WHAT IS CLAIMED IS:

1. A method of executing software, comprising:

identifying a first configuration information required to execute a first kernel code segment;

selecting a first accelerator set configured to execute said first kernel code segment; and

initiating a direct memory access transfer of said first configuration information to said first accelerator set.

2. The method of claim 1 further comprising building an entry in a kernel code execution table utilizing said first kernel code segment and said first configuration information

3. The method of claim 1, wherein said first configuration information is stored in a register and wherein the register defines a context.

4. The method of claim 1, further comprising identifying a first set of arguments.

5. The method of claim 4, wherein said initiating said direct memory access transfer further comprises transferring said first set of arguments.

6. The method of claim 1, further comprising identifying a first set of microcode.

7. The method of claim 6, wherein said initiating said direct memory access transfer further comprises transferring said first set of microcode.

8. The method of claim 1, further comprising identifying a second configuration information required to

execute a second kernel code segment and selecting a second accelerator set configured to execute said second kernel code segment

9. The method of claim 8, wherein said first accelerator set and said second accelerator set are overlapping.

10. The method of claim 10, wherein said first accelerator set and said second accelerator set are non-overlapping.

11. The method of claim 10, wherein said first kernel code segment executes on said first accelerator set and said second kernel code segment executes on said second accelerator set concurrently with the first .

12. The method of claim 11, wherein a third kernel code segment executes on said first accelerator set subsequent to said first kernel code segment and concurrently with said second kernel code segment.

13. The method of claim 1, further comprising identifying a second configuration information required to execute a second kernel code segment, wherein initiating said direct memory access includes transferring the second configuration information to said first accelerator set while said first kernel code set executes on said first accelerator set.

14. The method of claim 1, wherein initiating said direct memory access includes transferring a first set of

microcode to said first accelerator set while said first kernel code set executes on said first accelerator set.

15. The method of claim 1, further comprising retrieving a second kernel code segment wherein initiating said direct memory access includes transferring a first set of arguments to said first accelerator set while said first kernel code set executes on said first accelerator set.

16. The method of claim 1, further comprising determining completion requirements of said first kernel code segment to determine the order of execution of said first kernel code segment.

17. The method of claim 16, wherein said determining completion requirements of said first kernel code segment includes determining a variant of said first kernel code segment.

18. The method of claim 16, wherein said determining completion requirements of said first kernel code segment includes determining whether to execute said first kernel code segment in said first accelerator set or in a second accelerator set.

19. An apparatus, comprising:

a memory storing a set of configuration information;

an accelerator coupled to the memory; and

a kernel processor coupled to the memory, said kernel processor controlling the processing of at least one thread of program code on the accelerator by initiating a direct memory access transfer of the configuration information to the accelerator.

20. The apparatus of claim 19, further comprising at least one main processor coupled to the kernel processor and memory, least one main processor being configured to process overhead code.

21. The apparatus of claim 19, wherein said the accelerator is a multiple context processing element.

22. The apparatus of claim 21, wherein said multiple context processing elements are grouped into overlapping bins.

23. The apparatus of claim 21, wherein said multiple context processing elements are grouped into non-overlapping bins.

24. The apparatus of claim 23, wherein said kernel processor is configured to load a first kernel code segment into a first one of said non-overlapping bins and to load a second kernel code segment into a second one of said non-overlapping bins.

25. The apparatus of claim 19, wherein the accelerator is a digital signal processor.

26. The apparatus of claim 25, wherein the digital signal processor has a single instruction cache.

27. The apparatus of claim 25, wherein the digital signal processor has dual instruction caches.

28. The apparatus of claim 25, wherein the digital signal processor has an instruction cache configured with

dual-port memory, wherein a first port is coupled to a first bus and a second port is coupled to a second bus.

29. An apparatus configured to execute software, comprising:

means for identifying a first configuration information required to execute a first kernel code segment;

means for selecting a first accelerator set configured to execute said first kernel code segment; and

means for initiating a direct memory access transfer of said first configuration information to said first accelerator set.

30. A machine-readable medium having stored thereon instructions for processing elements, which when executed by said processing elements perform the following:

identifying a first configuration information required to execute a first kernel code segment;

selecting a first accelerator set configured to execute said first kernel code segment; and

initiating a direct memory access transfer of said first configuration information to said first accelerator set.